CLAIMS

What is claimed is:

1. A memory test circuit comprising:

a collar for coupling to a memory device for switching an address bus and a data bus of the memory device between an external circuit and the collar in response to a switching signal; and

a controller coupled to the collar for generating the switching signal, a test vector, and control signals on as few as seven control lines between the controller and the collar for testing the memory device with the test vector.

2. The memory test circuit of Claim 1 wherein the controller generates the control signals for multiple memory devices of various sizes.

3. The memory test circuit of Claim 1 wherein the control signals comprise a "CLEAR" signal, a "NEXT" signal, a "TEST_ENABLE" signal, a "START" signal, a "WRITE_ENABLE" signal, a "MOVE" signal, a "D" signal, a "MBIST_GO" signal, and a "TEST_IN" signal.

4. The memory test circuit of Claim 3 wherein the controller comprises logic for testing the memory device by performing the following functions:

(a) initializing parameters *m*, *n*, *memsize*, and *invert* wherein *n* is the maximum word size of the memory device, *m* is a power of two in the sequence (1, 2, 4, 8, ... 2^r) indicating a number of times a bit is repeated in the test

vector and *r* is a positive integer such that $2^r <= n <= 2^{r+1}$, *memsize* is a maximum address range of the memory device, *invert* is set to zero for generating the test vector having a bit pattern starting with zero and to one for generating test vector having a bit pattern starting with one;

(b) initializing a plurality of variables as follows: *isRead* to zero, *counter_address* to zero, *counter_n* to zero, and *counter_m* to one.

(c) initializing the "D" signal to zero;

(d) if *counter_m* >= *m*, transferring control to (e), otherwise transferring control to (f);

(e) setting *isM* to one and transferring control to (g);

(f) setting *isM* to zero;

(g) setting the "D" signal 122 to a previous value of the "D" signal 122 XOR *isM* XOR *invert*;

(h) if *counter_n* < n, transferring control to (i), otherwise transferring control to (j);

(i) setting the "MOVE" signal to one and transferring control to (k);

(j) setting the "MOVE" signal to zero ;

(k) setting the "NEXT" signal to the inverse of the "MOVE" signal;

(l) setting the "WRITE_ENABLE" signal to the inverse of the "MOVE" signal AND the inverse of *isRead*;

(m) if the "MOVE" signal equals one, then transferring control to (n), otherwise transferring control to (o);

(n) incrementing *counter_n* by one and transferring control to (r);

(o) if *isM* is equal to 1, then transferring control to (p), otherwise transferring control to (q);

(p) setting *counter_m* to one and transferring control to step 510.

(q) *counter_m* is incremented by one, and control transfers to (d);

(r) if *counter_address* >= *memsize*, then transferring control to (s), otherwise transferring control to (t);

(s) setting *isMemsize* to one and transferring control to (u);

(t) setting *isMemsize* to zero.

(u) setting the "CLEAR" signal to *isMemsize* OR the "START" signal;

(v) if *isMemsize* equals zero, then transferring control to (w), otherwise transferring control to (x);

(w) incrementing *counter_address* by one and transferring control to (d);

(x) if *isMemsize* AND *isread* equals zero, then transferring control to (y), otherwise transferring control to (z);

(y) setting *isRead* to one, setting *counter_address* to zero, and transferring control to (d); and

(z) setting the "TEST_OUT" signal to zero if an error is detected, otherwise setting the "TEST_OUT" signal to one.

5. The memory test circuit of Claim 3 further comprising logic in the controller for testing the memory device by performing the following functions:

(a) initializing parameters k, n, *memsize*, and *invert* wherein n is the maximum word size of the memory device, *memsize* is a maximum address range of the memory device, $k = 1,2,4,8,...,2^q$ wherein q is a positive integer such that

$2^q <= memsize < 2^{q+1}$, and *invert* is set to zero for generating the test vector having a bit pattern starting with zero and to one for generating test vector having a bit pattern starting with one;

(b) setting a variable *isRead* to zero;

(c) initializing a plurality of variables as follows: *counter_address* to zero, *counter_n* to one, and *counter_k* to zero;

(d) initializing the "D" signal to one;

(e) if *counter_k* equals *k*, then transferring control to (f), otherwise transferring control to (g);

(f) setting a variable *isK* to one and transferring control to (h);

(g) setting *isK* to zero;

(h) if *counter_n* equals *n*, then transferring control to (i), otherwise transferring control to (j);

(i) setting a variable *isN* to one and transferring control to (k);

(j) setting *isN* to zero;

(k) setting the "MOVE" signal to *isK* OR NOT *isN*;

(l) setting the "NEXT" signal to an inverse of the "MOVE" signal;

(m) setting the "WRITE_ENABLE" signal to the "NEXT" signal AND NOT *isRead;*

(n) setting the "D" signal to a previous value of the "D" signal XOR the "MOVE" signal XOR *invert;*

(o) if *isN* equals zero, then transferring control to (p), otherwise transferring control to (q);

(p) incrementing *counter_n* by one and transferring control to (r);

(q) if *isK* equals zero, then transferring control to (r), otherwise transferring control to (s);

(r) incrementing *counter_k* by one and transferring control to (t);

(s) setting *counter_k* to zero.

(t) if *counter_address* >= *memsize*, then transferring control to (u), otherwise transferring control to (v);

(u) setting a variable *isMemsize* to one and transferring control to (w);

(v) setting *isMemsize* to zero;

(w) setting the "CLEAR" signal to *isMemsize* OR the "START" signal;

(x) if *isMemsize* equals zero, then transferring control to (y), otherwise transferring control to (z).

(y) incrementing *counter_address* by one and transferring control to (e);

(z) if *isMemsize* AND *isread* equals zero, then transferring control to (aa), otherwise transferring control to (bb);

(aa) settng *isRead* to one and transferring control to (c); and

(bb) setting the "TEST_OUT" signal to zero if an error is detected, otherwise setting the "TEST_OUT" signal to one.

6. The memory test circuit of Claim 3 wherein the collar comprises a multiplexer coupled to the address bus of the memory device and the "TEST_ENABLE" signal.

7. The memory test circuit of Claim 6 wherein the collar comprises an address register coupled to the multiplexer.

8. The memory test circuit of Claim 7 wherein the address register is reset to zero by the "CLEAR" signal.

9. The memory test circuit of Claim 8 wherein the address register is incremented by the "NEXT" signal.

10. The memory test circuit of Claim 7 wherein the collar comprises an address comparator coupled to the address register for generating a memory enable signal.

11. The memory test circuit of Claim 6 wherein the collar comprises a data register coupled to the multiplexer for writing a test vector into the memory device.

12. The memory test circuit of Claim 11 wherein the test vector is transferred to the data register by the "MOVE" signal and the "D" signal.

13. The memory test circuit of Claim 11 wherein the collar comprises a data comparator coupled to the data register and to the memory device for generating the "TEST_IN" signal.

14. The memory test circuit of Claim 3 wherein the "MBIST_GO" signal has an initial value of one and is latched to zero upon detection of a memory device error.

15. A method for testing a memory device comprising:

switching an address bus and a data bus of the memory device between an external circuit and a collar in response to a switching signal; and

generating the switching signal, a test vector, and control signals on as few as seven control lines between a controller and the collar to test the memory device with the test vector.

16. The method of Claim 13 further comprising testing multiple memory devices of various sizes.

17. The method of Claim 13 wherein generating the switching signal, the test vector, and the control signals comprises generating a "CLEAR" signal, a "NEXT" signal, a "TEST_ENABLE" signal, a "START" signal, a "WRITE_ENABLE" signal, a "MOVE" signal, a "D" signal, a "MBIST_GO" signal, and a "TEST_IN" signal.

18. The method of Claim 17 wherein the switching signal is the "TEST_ENABLE" signal.

19. The method of Claim 17 wherein generating the switching signal, the test vector, and the control signals comprises performing the following functions:

(a) initializing parameters *m*, *n*, *memsize*, and *invert* wherein *n* is the maximum word size of the memory device, *m* is a power of two in the sequence (1, 2, 4, 8, ... 2^r) indicating a number of times a bit is repeated in the test vector and *r* is a positive integer such that $2^r <= n <= 2^{r+1}$, *memsize* is a maximum address range of the memory device, *invert* is set to zero for generating the test vector having a bit pattern starting with zero and to one for generating test vector having a bit pattern starting with one;

(b) initializing a plurality of variables as follows: *isRead* to zero, *counter_address* to zero, *counter_n* to zero, and *counter_m* to one.

(c) initializing the "D" signal to zero;

(d) if *counter_m* >= *m*, transferring control to (e), otherwise transferring control to (f);

(e) setting *isM* to one and transferring control to (g);

(f) setting *isM* to zero;

(g) setting the "D" signal 122 to a previous value of the "D" signal 122 XOR *isM* XOR *invert*;

(h) if *counter_n* < n, transferring control to (i), otherwise transferring control to (j);

(i) setting the "MOVE" signal to one and transferring control to (k);

(j) setting the "MOVE" signal to zero ;

(k) setting the "NEXT" signal to the inverse of the "MOVE" signal;

(l) setting the "WRITE_ENABLE" signal to the inverse of the "MOVE" signal AND the inverse of *isRead*;

(m) if the "MOVE" signal equals one, then transferring control to (n), otherwise transferring control to (o);

(n) incrementing *counter_n* by one and transferring control to (r);

(o) if *isM* is equal to 1, then transferring control to (p), otherwise transferring control to (q);

(p) setting *counter_m* to one and transferring control to step 510.

(q) *counter_m* is incremented by one, and control transfers to (d);

(r) if *counter_address* >= *memsize*, then transferring control to (s), otherwise transferring control to (t);

(s) setting *isMemsize* to one and transferring control to (u);

(t) setting *isMemsize* to zero.

(u) setting the "CLEAR" signal to *isMemsize* OR the "START" signal;

(v) if *isMemsize* equals zero, then transferring control to (w), otherwise transferring control to (x);

(w) incrementing *counter_address* by one and transferring control to (d);

(x) if *isMemsize* AND *isread* equals zero, then transferring control to (y), otherwise transferring control to (z);

(y) setting *isRead* to one, setting *counter_address* to zero, and transferring control to (d); and

(z) setting the "TEST_OUT" signal to zero if an error is detected, otherwise setting the "TEST_OUT" signal to one.

20. The method of Claim 17 wherein generating the switching signal, the test vector, and the control signals comprises performing the following functions:

(a) initializing parameters *k*, *n*, *memsize*, and *invert* wherein *n* is the maximum word size of the memory device, *memsize* is a maximum address range of the memory device, *k* = 1,2,4,8,...,2^q wherein q is a positive integer such that 2^q <= *memsize* < 2^{q+1}, and *invert* is set to zero for generating the test vector having a bit pattern starting with zero and to one for generating test vector having a bit pattern starting with one;

(b) setting a variable *isRead* to zero;

(c) initializing a plurality of variables as follows: *counter_address* to zero, *counter_n* to one, and *counter_k* to zero;

(d) initializing the "D" signal to one;

(e) if *counter_k* equals *k*, then transferring control to (f), otherwise transferring control to (g);

(f) setting a variable *isK* to one and transferring control to (h);

(g) setting *isK* to zero;

(h) if *counter_n* equals *n*, then transferring control to (i), otherwise transferring control to (j);

(i) setting a variable *isN* to one and transferring control to (k);

(j) setting *isN* to zero;

(k) setting the "MOVE" signal to *isK* OR NOT *isN*;

(l) setting the "NEXT" signal to an inverse of the "MOVE" signal;

(m) setting the "WRITE_ENABLE" signal to the "NEXT" signal AND NOT *isRead;*

(n) setting the "D" signal to a previous value of the "D" signal XOR the "MOVE" signal XOR *invert;*

(o) if *isN* equals zero, then transferring control to (p), otherwise transferring control to (q);

(p) incrementing *counter_n* by one and transferring control to (r);

(q) if *isK* equals zero, then transferring control to (r), otherwise transferring control to (s);

(r) incrementing *counter_k* by one and transferring control to (t);

(s) setting *counter_k* to zero.

(t) if *counter_address* >= *memsize*, then transferring control to (u), otherwise transferring control to (v);

(u) setting a variable *isMemsize* to one and transferring control to (w);

(v) setting *isMemsize* to zero;

(w) setting the "CLEAR" signal to *isMemsize* OR the "START" signal;

(x) if *isMemsize* equals zero, then transferring control to (y), otherwise transferring control to (z).

(y) incrementing *counter_address* by one and transferring control to (e);

(z) if *isMemsize* AND *isread* equals zero, then transferring control to (aa), otherwise transferring control to (bb);

(aa) settng *isRead* to one and transferring control to (c); and

(bb) setting the "TEST_OUT" signal to zero if an error is detected, otherwise setting the "TEST_OUT" signal to one.

21. The method of Claim 20 further comprising initially setting the "MBIST_GO" signal to one and latching

the "MBIST_GO" signal if to zero if the "TEST_OUT" signal is set to zero upon detection of a memory device error.